Felix Design Report:

Reporter：Noah tingkaihu Tom Felix

Procedural Modeling of a Traditional Chinese Cookie (Peach Crisp)

Peach crisp is a traditional Chinese pastry known for its crispy texture, handcrafted pressed shape, and classic almond decoration. The goal of this project is to use procedural modeling techniques in OpenSCAD to create a realistic 3D model of a peach crisp. By carefully designing the shape, texture, and decoration, the model aims to not only replicate the traditional appearance of the cookie but also leverage computational generation techniques to highlight its unique characteristics.

The modeling process consists of three main components:

1. Cookie Base – A scaled ellipsoid is used to simulate the round shape of the peach crisp. A Boolean difference operation creates a central depression, mimicking the hand-pressed indentation commonly found in traditional peach crisps.
2. Almond Decoration – Multiple spheres are combined using the hull() function to generate a smooth, oval-shaped almond, accurately representing the classic garnish on the cookie.
3. Cream Topping – A semi-transparent sphere is used to simulate the cream layer, adding a realistic touch and enhancing the visual appeal of the model.

Cookie Base (Main Body)

The base of the cookie is modeled as an ellipsoid using the scale() function to adjust the proportions. The indentation in the center is achieved through a difference() operation, subtracting a smaller, flattened sphere from the main body. This technique effectively replicates the pressed shape commonly seen in handmade peach crisps.

Almond Decoration

The almond is created using the hull() function to connect multiple spheres, forming a smooth and organic shape. The individual spheres vary in size, and their positions are carefully adjusted to ensure a natural almond-like appearance. The almond is then placed on top of the cookie for a realistic finishing touch.

Cream Topping

The cream component is a semi-transparent ellipsoid, achieved by defining an RGBA color with reduced opacity. This layer adds complexity to the visual representation of the cookie and enhances its realism.

This project successfully demonstrates how procedural modeling can be used to create realistic representations of traditional food items. By leveraging Boolean operations, transformations, and surface blending, the model effectively captures the defining characteristics of a peach crisp. The process also highlights the power of computational design in food visualization, offering potential applications in digital art, 3D printing, and virtual simulations.

Future improvements could include texture mapping for enhanced realism and the addition of more detailed surface variations to better mimic the handmade quality of a real peach crisp.

Further illustration to code:

```
module cookie() {
    base_radius = 100;      // The radius of the cookie
    height = 50;            // The thickness of the cookie
    smoothness = 100;       // Smoothness (controls the number of faces of the sphere)
```

```
    color([0.8, 0.6, 0])    // Set the color to brownish-yellow (cookie color)
    difference() {    // Use the difference() function to perform a Boolean difference operation (subtract a part)

        // Generate the main body of the cookie (ellipsoid)
        scale([1, 1, height / (2 * base_radius)])
        sphere(r = base_radius, $fn = smoothness);

        // Carve a shallow depression in the center of the cookie
        translate([0, 0, height * 0.6])
        scale([1, 1, 0.3])
        sphere(r = base_radius * 0.5, $fn = smoothness);
    }
}

module cream() {
    base_radius = 60;    // The radius of the cream (smaller than the cookie)
    height = 49;             // The thickness of the cream (slightly thinner)
    smoothness = 100;    // Smoothness

    color([0.8, 0.1, 0.1, 0.85]) // Set the color to red with semi-transparency (cream)
    difference() {
        // Generate the cream shape (ellipsoid)
        scale([1, 1, height / (2 * base_radius)])
        sphere(r = base_radius, $fn = smoothness);
    }
}
```

```
module almond() {
    color([0.8, 0.6, 0.4]) // The color of the almond (light brown)
    difference() {
        hull() {    // Use the hull() function to create a smooth convex shape
            adjustable_ball1(15, [0, 0, 20]);      // Large sphere
            adjustable_ball2(9, [10, 10, 30]);    // Medium sphere
            adjustable_ball3(3, [20, 20, 40]);    // Small sphere
        }
    }
}
```

ZiRui Explanation to code

The color() function is used to set the color of the object. The sphere(20) function first creates a spherical shape, which is then flattened along the Z-axis using scale([1,1,0.4]). The cylinder(h=3, d=60) function creates the base of the plate. The hull() function is used to connect two cylinders to form the slightly raised outer rim of the plate. The translate([0, 0, 3]), cylinder(h=2, d=53) operation subtracts the middle part, creating the concave effect of the plate.

We separated the three models into different parts, each with distinct colors, and applied colors accordingly before assembling them in Three.js. We used STLLoader to load .stl format 3D model files.

The scene.add(dorayakiGroup); function sets the initial position of the dorayaki.

The following code makes the cookie rotate around its Y-axis while also orbiting around the cake:

```
let cookieRadius = 60;

cookieGroup.position.x = Math.cos(elapsedTime) * cookieRadius;

cookieGroup.position.z = Math.sin(elapsedTime) * cookieRadius;

cookieGroup.rotation.y += 0.05;
```

Similarly, the following code makes the dorayaki rotate around its X-axis while also orbiting around the cake:

```
let dorayakiRadius = 60;

dorayakiGroup.position.x = Math.cos(-elapsedTime) * dorayakiRadius;

dorayakiGroup.position.z = Math.sin(-elapsedTime) * dorayakiRadius;

dorayakiGroup.rotation.x += 0.05;
```

TingKai Hu explanation

```
$fn = 100; // Makes the model smoother

module cake_base() {
    color("saddlebrown") // Set the color to brown
    cylinder(h=8, r=20); // Create a cylindrical cake base
```

```
}

module cream_layer() {
    translate([0, 0, 8]) // Move it above the cake base
    color("white") // Set the cream color to white
    cylinder(h=2, r=20);
}

module strawberry() {
    color("red") // Set the strawberry color to red
    scale([1, 1, 1.5]) // Stretch the sphere to form an irregular shape
    sphere(3.5); // Create an irregular sphere for the strawberry

    for (i = [0:60:360]) { // Place 6 leaves on the strawberry
        rotate([0, 0, i]) // Rotate around the Z-axis to distribute leaves evenly
        translate([0, 0, 5]) // Move them to the top of the strawberry
        rotate([0, 30, 0]) // Tilt the leaves by rotating them 30 degrees along the Y-axis
        color("green") // Set the leaf color to green
        cylinder(h=1.2, r1=0, r2=2.5); // Create a cone-shaped leaf with a small top radius and a larger base
    }
}

module strawberries_go() { // Position the strawberries
    for (i = [0:60:360]) { // Place 6 strawberries
        rotate([0, 0, i]) // Rotate around the Z-axis to position them evenly
        translate([14, 0, 10]) // Move them to the upper edge of the cake
```

```
        rotate([0, 90, 0]) // Rotate along the Y-axis to lay them down
        strawberry(); // Load the strawberry module
    }
}

module chocolate() {
    translate([0, 0, 10]) // Move the chocolate piece to the top
    color("pink") // Set the color to pink
    cylinder(h=4, r1=3, r2=0); // Create a pointed pink chocolate decoration
}

module cake() { // The cake module combines all the components
    cake_base();
    cream_layer();
    strawberries_go();
    chocolate();
}

cake(); // Generate the cake
```

Noah’s code and further illustation to it

```
$fn = 100; // Makes the model smoother

module cake_base() {
```

```
    color("peru") // Set the base color to light brown
    cylinder(h=8, r=22); // Create the cylindrical base of the cake
}

module cream_layer() {
    translate([0, 0, 8]) // Position the cream layer above the base
    color("ivory") // Set the cream color to ivory
    cylinder(h=3, r=22);
}

module blueberry() {
    color("blue") // Set the blueberry color to blue
    sphere(3); // Create a spherical blueberry

    for (i = [0:120:360]) { // Add 3 small bumps on the blueberry
        rotate([0, 0, i]) // Rotate them around the Z-axis
        translate([2, 0, 2]) // Position them slightly above the surface
        scale([0.5, 0.5, 0.5]) // Scale down the spheres for a natural look
        sphere(1);
    }
}

module blueberries_go() { // Arrange blueberries on top of the cake
    for (i = [0:45:360]) { // Place 8 blueberries around the edge
        rotate([0, 0, i]) // Rotate each one to an even position
        translate([16, 0, 11]) // Move them to the top edge of the cake
        blueberry(); // Load the blueberry model
```

```
    }
}

module chocolate_shavings() {
    translate([0, 0, 11])
    color("sienna") // Set the color to brown
    for (i = [0:90:360]) { // Create four chocolate shavings
        rotate([0, 0, i]) // Spread them around the center
        translate([5, 0, 0]) // Offset each one slightly
        rotate([0, 20, 0]) // Tilt them for a natural effect
        cylinder(h=3, r1=0.5, r2=2.5); // Create a cone-shaped shaving
    }
}

module cake() { // Combine all elements to form the cake
    cake_base();
    cream_layer();
    blueberries_go();
    chocolate_shavings();
}

cake(); // Generate the cake
```